2019 20th International Conference on Research and Education in Mechatronics (REM)
May 23-24, 2019, Wels, Austria

PSO-based Algorithm for Inverse Kinematics Solution of Robotic Arm Manipulators

Malek Alkayyali
Mechatronics Engineering Department
Philadelphia University
Jerash, Jordan
malekkayyali@gmail.com

Tarek A. Tutunji
Mechatronics Engineering Department
Philadelphia University
Jerash, Jordan
ttutunji@philadelphia.edu.jo

***Abstract*—Inverse kinematics is an important and challenging problem in the robotics field. The analytical solution of the problem is complex and time consuming. In this paper, particle swarm optimization (PSO) is used to solve the inverse kinematics problem of 6-DOF KUKA KR 6 R900 robotic arm manipulator. The proposed algorithm uses PSO to generate inverse kinematic solutions by searching the six-dimensional joint space. These solutions are then used to construct the end-effector pose using forward kinematics with the objective of finding the optimal joint angles. A multi-objective fitness function is used to measure the position and orientation errors of the generated solutions. Simulation results for three different cases show that the proposed PSO-based algorithm can solve the inverse kinematics problem efficiently with minimum error in the end-effector pose.**

Keywords—Inverse kinematics; Robotic arm manipulator; 6-DOF; KUKA KR 6 R900; Particle Swarm Optimization (PSO); Optimization

I. INTRODUCTION

Robotic arm manipulator is defined as a chain of links connected by joints which give the rigid bodies (i.e. links) the ability to move freely. Inverse kinematics is concerned with computing the required joints variables which correspond to a given end-effector pose. Inverse kinematics problem is more complex than the forward kinematics problem due to the nonlinearity of its equations, existence of multiple or infinite number of solutions, or even no solution due to singularities of the problem [1].

In the recent years, optimization techniques have become very popular in solving engineering problems, because of the advancement in programming applications and software. All optimization algorithms aim to minimize or maximize a cost function to get the optimal solution for a given problem. Particle swarm optimization (PSO) was introduced by Kennedy and Eberhart. This algorithm attempts to mimic the behavior of swarm in nature such as insects and animals schooling [2].

Due to the difficulties and limitations of conventional inverse kinematics, different optimization and artificial intelligence algorithms were used in literature to solve the inverse kinematics problem of robotic arm manipulators. In [3], eight soft computing algorithms were used to solve the inverse kinematics problem for a single point and a whole path. The simulation results proved that particle swarm optimization could be used to solve the inverse kinematics problem efficiently. In [4], a modified Artificial Bee Colony (ABC) algorithm was proposed. Knowledge-based Artificial Bee Colony (K-ABC) was used to solve the inverse kinematics problem of 5 degrees of freedom (DOF) robotic arm manipulator. The simulation results showed that PSO performs better than K-ABC in solving helical path for the 5-DOF manipulator.

In [5], three recurrent neural networks were trained to solve the inverse kinematics problem of 6-DOF Stanford robotic arm manipulator. A hybrid algorithm with Binary Coded Genetic Algorithm (BCGA) was proposed to increase the accuracy of the output of the recurrent neural networks. The results showed a significant improvement in the accuracy of the output link angles. The algorithm reduced the error in end-effector position error.

PSO algorithm was used in different researches to solve the inverse kinematics problem of robotic arm manipulators. Five variants of PSO were used in [6] to solve the inverse kinematics problem of 2-DOF double link articulated system using forward kinematics equations. The simulation results proved that inertia weight PSO performed better than the other four versions of PSO. Therefore, it can be concluded that PSO is a suitable optimization technique that can be used to solve the inverse kinematics problem of robotic arm manipulator. However, the authors did not test their algorithm on manipulators with higher degrees of freedom.

In [7], two versions of particle swarm optimization algorithm were used to solve the inverse kinematics problem of redundant manipulators with 9 to 180 DOF. The proposed algorithm used a fitness function that prevented the robot from self-collision. It also relied on the error of the position and orientation of the end-effector. The simulation results showed that Constriction Factor PSO performed better than the probabilistic-based Bare Bone PSO in solving the inverse kinematics problem of robotic manipulators with 30 DOF and higher. However, their work was tested using SuperBot robot which has different architecture and configuration than KUKA robots.

An algorithm using particle swarm optimization was proposed in [8] to solve the inverse kinematics problem of 7-

978-1-5386-9257-8/19/$31.00 ©2019 IEEE

DOF redundant arm manipulator. The proposed fitness function relied on the error in position and joints angle displacement. The simulation results showed that PSO was able to solve the inverse kinematics of redundant manipulator successfully. However, the proposed fitness function did not include the error in orientation of the end-effector which could be included to improve the solution of the inverse kinematics problem.

Inverse kinematics problem of redundant manipulator was also studied in literature [9, 10]. In both cases, closed form solutions were used to solve the inverse kinematics problem. This involves solving complex mathematical calculations which is time consuming and can be overcome by using optimization algorithms.

In this paper, a PSO-based inverse kinematics solution for robotic arm manipulators is proposed. The proposed algorithm takes into account the error in the position and orientation of the end-effector of the manipulator. The algorithm is used to solve the inverse kinematics problem of single pose for KUKA KR 6 R900 robotic arm manipulator.

The rest of the paper is organized as follows. In section II, the forward kinematics model of KUKA KR6 R900 and particle swarm optimization (PSO) algorithm are illustrated. The research methodology is given in Section III and simulation results for three cases are presented in Section IV. Finally, conclusions and recommendations for future work are given in Section V.

II. Theoritical Background

This section provides the basic theoretical and mathematical background of manipulator kinematics and PSO optimization.

A. Kinematics Model

Forward kinematics is concerned in finding the pose of the end-effector of robotic arm manipulator with respect to a fixed frame in space given the value of each joint variable (i.e. the angle of revolute joints and translation of prismatic joints). In order to find the pose of the end-effector, classical Denavit-Hartenberg (DH) convention is used to find the homogenous transformation matrix between the end-effector frame and the base frame by affixing appropriate frames on each joint then multiplying the transformation matrices between each two consecutive joints. The classical DH-transformation matrix between two frames is given by (1) [1],

$$\boldsymbol{T_i^{i-1}} = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

where: c is cosine function, s is sine function, a_i and α_i are the i[th] link length and angle, d_i and θ_i are the i[th] joint offset and angle.

Six degrees of freedom (6-DOF) KUKA KR 6 R900 sixx arm manipulator is used to test the proposed method to solve the inverse kinematics problem. Fig. 1 shows a side view of the robotic manipulator and the coordinate frames of each joint while Table I shows the KUKA dimensions [11]. The DH parameters for the manipulator are presented in Table II. These parameters were used to build the forward model in the simulation runs.

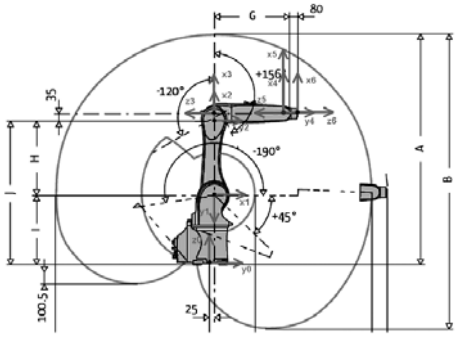

Fig. 1. KUKA KR 6 R900 sixx joints coordinate frames [11]

TABLE I. Dimensions of KUKUA KR 6 R900 sixx [11]

A [mm]	B [mm]	G [mm]	H [mm]	I [mm]	J [mm]
1276	1620	420	455	400	855

TABLE II. DH parameters for KUKA KR 6 R900 sixx [11]

i	1	2	3	4	5	6
a_i [mm]	25	H	35	0	0	0
α_i [rad]	-π/2	0	π/2	-π/2	π/2	π
d_i [mm]	I	0	0	-G	0	-80
θ_i [rad]	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6

The final homogenous transformation matrix of the end-effector frame with respect to the base frame ($\boldsymbol{T_e^b}$) can be found by substituting the DH parameters from Table II in equation (1) and multiplying the resultant six matrices. The final homogenous transformation matrix was used with PSO algorithm to optimize the joint variables that corresponds to the desired end-effector pose.

B. Particle Swarm Optimization (PSO)

PSO is a naturally inspired algorithm that mimics the swarm behavior in nature such as bird and fish schooling. PSO algorithm uses a swarm consists of (N) particles that search in a space of dimension (D) by moving the trajectories of individual particles to obtain the best solution according to a certain quality measure (i.e. fitness function). Each particle has its own position and velocity which are given by the vectors in equations (2) and (3), respectively [2].

$$\boldsymbol{X_i} = [x_1\ x_2\ \dots\ x_D]\ ; i = 1, \dots, N \quad (2)$$

$$\boldsymbol{V_i} = [v_1\ v_2\ \dots\ v_D]\ ; i = 1, \dots, N \quad (3)$$

PSO algorithm starts by initializing the swarm with N-random particles (i.e. vectors) which represent N-random solutions in space. Then, each particle starts to modify its position according to its fitness as well as the fitness of other particles. The fitness of every particle in the swarm is initially calculated, then the particle with the best fitness is assumed to be the personal best particle ($\boldsymbol{P_{best}}$) and the global best particle ($\boldsymbol{g_{best}}$). In the following iterations, the velocity and position of the

particles are updated each iteration according to (4) and (5), respectively [12]. The fitness of the new personal best particle is compared with the previous global best particle fitness to find the new global best position which has the smallest value of fitness so far. This process is repeated until one termination condition is met.

$$\boldsymbol{V_i}(iter) = K\big[w\,\boldsymbol{V_i}(iter-1) + c_1\,r_1\,\big(\boldsymbol{P_{best}} - \boldsymbol{X_i}(iter-1)\big) + c_2\,r_2\,\big(\boldsymbol{g_{best}} - \boldsymbol{X_i}(iter-1)\big)\big] \quad (4)$$

$$\boldsymbol{X_i}(iter) = \boldsymbol{X_i}(iter-1) + \boldsymbol{V_i}(iter) \quad (5)$$

Where:

$K = \frac{2\times rand}{\left|2-\varphi-\sqrt[2]{\varphi^2-4\times\varphi}\right|}$: $\varphi = c_1\,r_1 + c_2\,r_2$

$w = w_{max} - \frac{(w_{max}-w_{min})}{iter_{max}} \cdot iter$

r_1, r_2: Random real numbers

c_1, c_2: Random integer numbers

This is a PSO variant that improves the new particle using weight inertia (w) and constriction factor (K) [12]. Integers c_1 and c_2 determine the relative influence of the social and cognitive components, while the constriction coefficient (K) minimizes the explosive feedback effects [13]. The inertia factor (w) makes a balance between the global and local particles exploration abilities and *iter* is the sequential number of the current iteration [14]. In this work, w_{max} and w_{min} were set to 1 and 0, respectively. Also, the range used for r_1 and r_2 was between 0 and 1 while the range used for c_1 and c_2 was 2 to 4.

This PSO variant was used in previous work to design the optimal integer-order PID and fractional-order PID (FOPID) controllers for different plants [15]. It showed promising results in solving such complex problems because it combines the advantages of both versions of PSO algorithms which are: Constriction factor PSO and Inertia weight PSO. Therefore, this PSO variant is used to solve the inverse kinematics problem of 6-DOF KUKA KR 6 R900 manipulator.

One crucial step when using any optimization algorithm is stating an appropriate fitness function which measures the closeness of the solution to achieve some desired requirements. In this research, a fitness function that relies on the 2-norm error of the position and orientation of the end-effector is adopted from [3]. PSO algorithm is used to find the six joints angles which correspond to a desired pose of the end-effector by minimizing the fitness function given by

$$E_P = \frac{\|\boldsymbol{P_{PSO}} - \boldsymbol{P_d}\|_2}{\|\boldsymbol{P_d}\|_2} \quad (6)$$

$$E_R = \frac{\|\boldsymbol{R_{PSO}} - \boldsymbol{R_d}\|_2}{\|\boldsymbol{R_d}\|_2} \quad (7)$$

$$F = \rho E_p + (1-\rho)E_R \quad (8)$$

$$\rho = \alpha e^{-\sigma} + \beta : \sigma = E_R\,, \alpha = 0.7\,, \beta = 0.3 \quad (9)$$

where:

$\boldsymbol{P_{PSO}}$, $\boldsymbol{R_{PSO}}$ are the calculated position vector and rotation matrix using PSO.

$\boldsymbol{P_d}$, $\boldsymbol{R_d}$ are the desired position vector and rotation matrix.

The adopted fitness function has an inertia factor (ρ) that shows the importance of each term in the fitness function, it will be large if the error in orientation is small which allows the error in position a higher contribution to the fitness function [3].

The proposed algorithm in this work uses PSO to search a 6-dimenssional space for the optimal joints variables (θ_i). Then, forward kinematics equations are used to calculate the end-effector pose. This pose is compared with desired pose using the fitness function (8). In this case, the particle vector is the joints angles as

$$\boldsymbol{q} = [\theta_1\ \theta_2\ \theta_3\ \theta_4\ \theta_5\ \theta_6] \quad (10)$$

Fig. 2 shows the proposed algorithm that is used to solve the inverse kinematics problem using PSO.

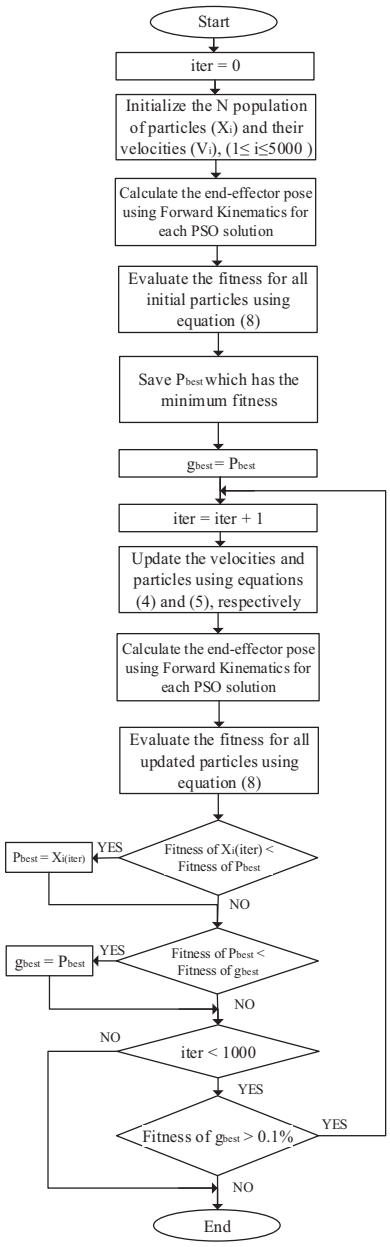

Fig. 2. Proposed algorithm flowchart

III. Research Methodology

In this paper, an algorithm using PSO technique is used to solve the inverse kinematics problem of KUKA KR 6 R900 sixx arm manipulator. The problem was solved by searching a 6-dimensional space for the optimal joints angles that give the desired position and orientation. The quality of PSO solutions were measured using a fitness function that relies on the error in pose of the end-effector which is given by equation (9). Fig. 3 shows a block diagram for the proposed algorithm where the difference between the desired and PSO transformation matrices ($\boldsymbol{E}$) is used to extract the 2-norm errors described by equations (6) and (7).

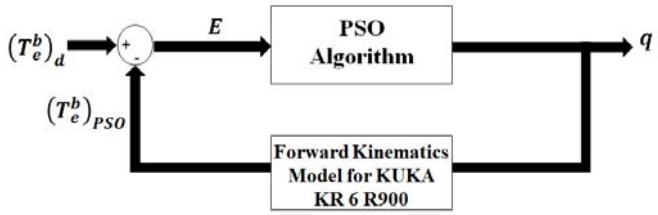

Fig. 3. Block diagram of the proposed algorithm

The proposed algorithm was coded using MATLAB script file. The algorithm was used to solve the inverse kinematics problem for KUKA KR 6 R900 sixx arm manipulator for different cases.

Three different end-effector poses were used to evaluate the performance of the proposed algorithm in solving the inverse kinematics problem. In each case, the global best particles which represent the optimal joints angles were plotted versus the iterations. Also, the corresponding fitness value for each global best particle was plotted versus the iterations. Moreover, the solutions obtained using the proposed algorithm were compared with MATLAB robotics toolbox.

IV. Simulation Results

The proposed algorithm was tested to solve the inverse kinematics problem of KUKA KR 6 R900 for three different desired poses. The results were compared to Peter Corke MATLAB robotics toolbox that uses pseudo-inverse Jacobian algorithm [16].

The proposed algorithm takes into consideration the mechanical limitation of the six joints angles. Table III shows the higher and lower limits of the joints angles obtained from KUKA KR 6 R900 sixx datasheet [17].

TABLE III. Joints angles ranges [17]

Joint	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
Max. limit [degrees]	170	45	156	185	120	350
Min. limit [degrees]	-170	-190	-120	-185	-120	-350

A. Case One

The desired end-effector pose is shown in Fig. 4 and is given in equation (11).

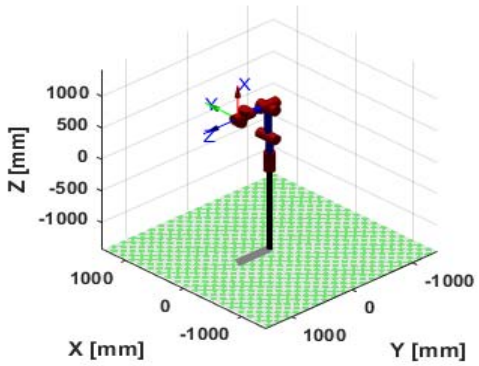

Fig. 4. The desired end-effector pose (Case One)

$$(T_e^b)_d = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 525 \\ 1 & 0 & 0 & 890 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (11)$$

The joints angles obtained using the proposed algorithm are given in (12). Fig. 5 shows the global best solutions versus iterations while Fig. 6 shows the fitness of global particles versus iterations. It can be shown from Fig. 5 that PSO converges to the optimal joints angles that correspond to the solution of the inverse kinematics very quickly (i.e. after 15 iterations). Moreover, Fig. 6 shows the optimal solution has a fitness of 0.06562% which is less than the desired fitness value.

$$\boldsymbol{q} = [90.2^o \quad -90^o \quad 0^o \quad 103.8^o \quad 1.2^o \quad 255.8^o] \quad (12)$$

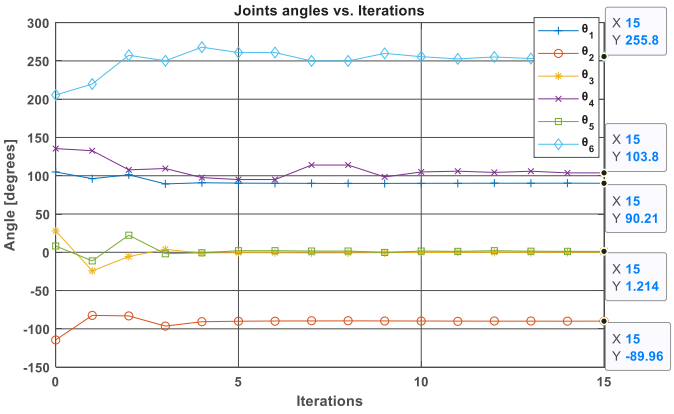

Fig. 5. Joints angles vs. iterations (Case One)

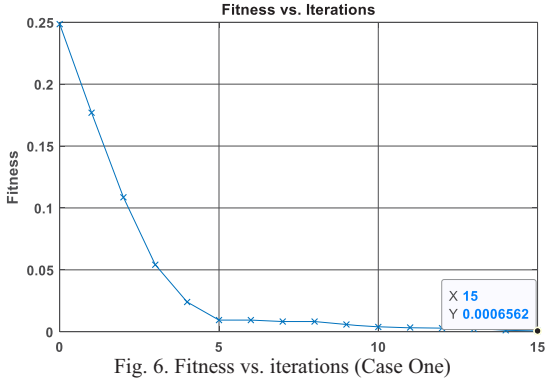

Fig. 6. Fitness vs. iterations (Case One)

MATLAB robotics toolbox was unsuccessful in solving the inverse kinematics problem for this case. The algorithm uses initial joints angles and therefore the convergence depends on these initial values. Several trials with different initial values were tested.

B. *Case Two*

The desired end-effector pose is shown in Fig. 7, it is also given by equation (13).

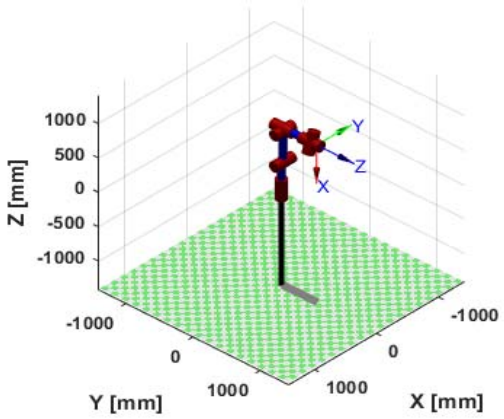

Fig. 7. The desired end-effector pose (Case Two)

$$\left(\boldsymbol{T}_{\boldsymbol{e}}^{\boldsymbol{b}}\right)_{\boldsymbol{d}} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 525 \\ -1 & 0 & 0 & 890 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (13)$$

The joints angles obtained using the proposed algorithm are given in (14). Fig. 8 shows the global best solutions versus iterations while Fig. 9 shows the fitness of global particles versus iterations. It can be shown from Fig. 8 that PSO converges to the optimal joints angles that correspond to the solution of the inverse kinematics after performing 16 iterations. Moreover, Fig. 9 shows the optimal solution has a small relative error which corresponds to a fitness of 0.03911% which is less than the desired fitness value.

$$\boldsymbol{q} = [90.1^o \quad -90^o \quad 0.1^o \quad -71.7^o \quad -0.8^o \quad 251.8^o] \quad (14)$$

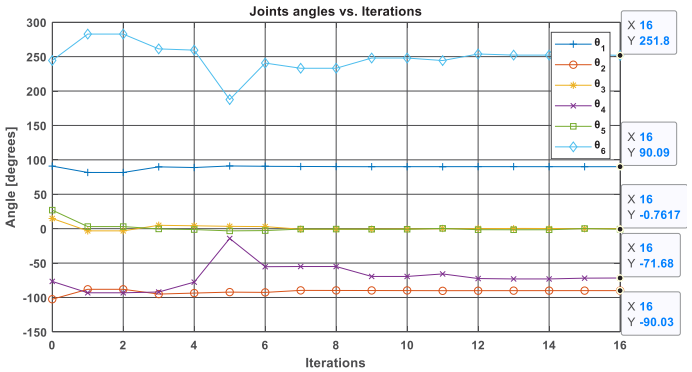

Fig. 8. Joints angles vs. iterations (Case Two)

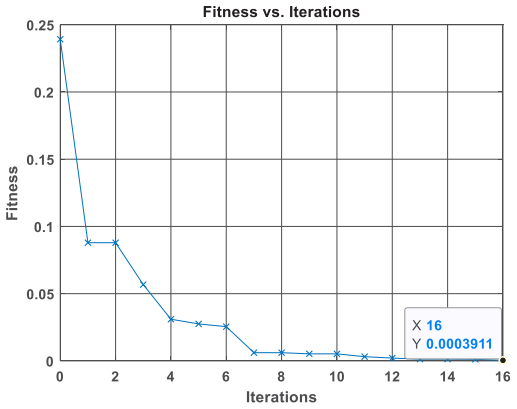

Fig. 9. Fitness vs. iterations (Case Two)

MATLAB robotics toolbox was used to solve the inverse kinematics problem for the desired pose given in (13). The algorithm solved the problem and the resultant joints angles are given in (15). This solution produced a better fitness of 0.0215% but it can be seen that the third joint angle ($\theta_3 = -170.5$) lied outside the range specified in the datasheet of KUKA KR 6 R900 sixx. However, the proposed algorithm does not produce a solution outside the joints angles limits because the algorithm

$$\boldsymbol{q} = [90^o \quad -8.8^o \quad -170.5^o \quad 180^o \quad -89.3^o \quad 0^o] \quad (15)$$

takes into consideration these limitations.

C. *Case Three*

The desired end-effector pose is shown in Fig. 10, it can be given by the transformation matrix in (16).

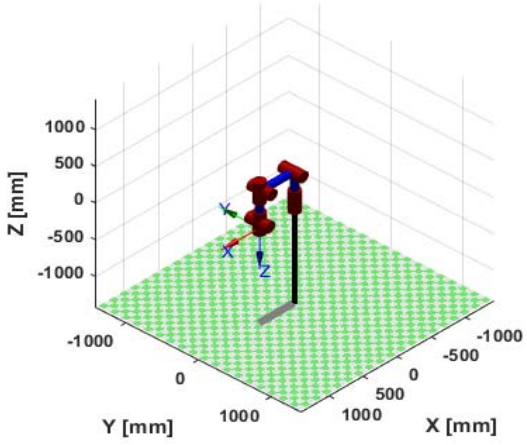

Fig. 10. The desired end-effector pose (Case Three)

$$\left(\boldsymbol{T}_{\boldsymbol{e}}^{\boldsymbol{b}}\right)_{\boldsymbol{d}} = \begin{bmatrix} 1 & 0 & 0 & 515 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -100 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (16)$$

The joints angles obtained using the proposed algorithm are given in (17). Fig. 11 shows the global best solutions versus iterations. While Fig. 12 shows the fitness of global particles

versus iterations. Fig. 11 shows that PSO converges to the optimal joints angles that correspond to the solution of the inverse kinematics after performing 16 iterations. Moreover, Fig. 12 shows the optimal solution has a small relative error which corresponds to a fitness of 0.02295% which is less than the desired fitness value.

$$\boldsymbol{q} = [90.2^o \quad -90^o \quad 0^o \quad 103.8^o \quad 1.2^o \quad 255.8^o] \qquad (17)$$

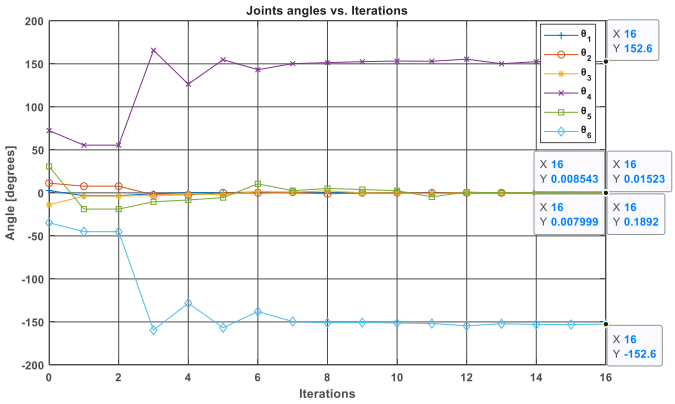

Fig. 11. Joints angles vs. iterations (Case Three)

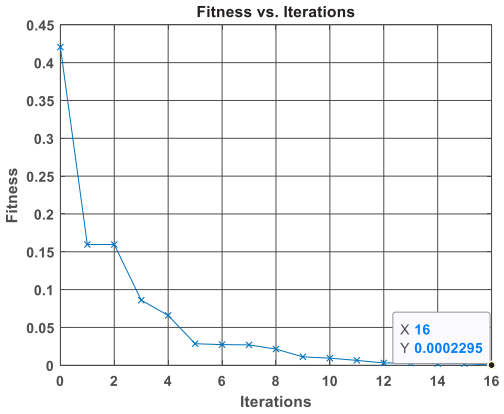

Fig. 12. Fitness vs. iterations (Case Three)

MATLAB robotics toolbox was used to solve the inverse kinematics problem for the desired pose given in (16). The algorithm converged to the solution shown in equation (18) which has a fitness of 0%. However, this is a trivial problem as the pose corresponds to the home position.

$$\boldsymbol{q} = [0^o \quad 0^o \quad 0^o \quad 0^o \quad 0^o \quad 0^o] \qquad (18)$$

V. Conclusions and Future Work

This paper proposed a PSO-based inverse kinematics algorithm for robotic arm manipulators. The paper examined the effectiveness of using PSO technique to solve the inverse kinematics problem for 6-DOF KUKA KR 6 R900 sixx manipulator. It was shown that PSO is a suitable optimization technique that could be used to solve such complex problems.

The algorithm was validated using three different simulated cases and the results were compared with MATLAB robotics toolbox. In all cases, PSO was able to converge to the optimal joints angles (i.e. that correspond to a desired end-effector pose) in less than 16 iterations with small normalized error in position and orientation (i.e. less than 0.1%).

The proposed algorithm could be extended to solve the inverse kinematics problem for a path. The algorithm would be very effective in solving trajectory planning problems because it can avoid sudden changes in joint speeds and singularity points. Moreover, the algorithm could be used effectively with different arm manipulator models.

References

[1] Siciliano, B. Robotics: Modelling, Planning and Control. Springer-Verlag New York, LLC, 2008.

[2] Xin-she., Y. Nature-inspired Optimization Algorithms. Elsevier, 2014.

[3] Carlos Lopez-Franco, Jesus Hernandez-Barragan, Alma Y. Alanis, Nancy Arana-Daniel. A soft computing approach for inverse kinematics of robot manipulators. *Engineering Applications of Artificial Intelligence*, 74, 104-120, 2018.

[4] Ahmed El-Sherbiny, Mostafa A. Elhosseini, Amira Y. Haikal. A new ABC variant for solving inverse kinematics problem in 5 DOF robot arm. *Applied Soft Computing*, 73, 24-38, 2018.

[5] Raşit Köker. A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization. *Information Sciences*, 528-543, 2013.

[6] Adel. M. Alimi, Nizar Rokbani. Inverse Kinematics Using Particle Swarm Optimization, A Statistical Analysis. *International Conference on Design and Manufacturing (IConDM)*: Procedia Engineering, 64. 1602-1611, 2013.

[7] Thomas Joseph Collinsm, Wei-Min Shen. Particle Swarm Optimization for High-DOF Inverse Kinematics. *3rd International Conference on Control, Automation and Robotics:* IEEE, 2017.

[8] Hsu-Chih Huang, Chien-Po Chen, Pei-Ru Wang. Particle Swarm Optimization for Solving the Inverse Kinematics of 7-DOF Robotic Manipulators. *IEEE International Conference on Systems, Man, and Cybernetics*: IEEE, 64. 3105-3110, 2012.

[9] Isiah Zaplana, Luis Basanez. A novel closed-form solution for the inverse kinematics of redundant manipulators through workspace analysis. *Mechanism and Machine Theory*, 121, 829-843, 2018.

[10] Martin Pfurner, 2016. Closed form inverse kinematics solution for a redundant anthropomorphic robot arm. *Computer Aided Geometric Design*, 47, 163-171, 2018.

[11] Paweł Szulczyn´ Ski, Krzysztof Kozłowski. Parametric programming of industrial robots. *Archives of Control Sciences*, 215-225, 2012.

[12] Yonghe Lu, Minghui Liang, Zeyuan Ye, Lichao CaoSchool. Improved particle swarm optimization algorithm and its application in text feature selection. *Applied Soft Computing*, 629-636, 2015.

[13] M. Clerc, J. Kennedy. The Particle Swarm—Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Transactions on Evolutionary Computation*, 58-73, 2002.

[14] Y. Shi, R. Eberhart. A Modified Particle Swarm Optimizer. *IEEE International Conference on Evolutionary Computation Proceedings, IEEE World Congress on Computational Intelligence*, 58-73, 1998.

[15] Zaer S. Abo-Hammour, Malek Alkayyali, Hussam J. Khasawneh, Mohammad I. Al Saaideh. On The Design of the Integer and Fractional PID Controllers Using Particle Swarm Optimization. *The International Conference on Fractional Differentiation and its Applications (ICFDA)*: Digital Conference Proceedings SSRN, 2018.

[16] Peter I. Corke. Robotics TOOLBOX for MATLAB. CSIRO Manufacturing Science and Technology, Pinjarra Hills, AUSTRALIA, 2001.

[17] KUKA Robots KR AGILUS sixx With W and C Variants Specification Datasheet, 2018